Introduction

In this example, the individual steps of the DynaDock procedure as implemented in the DynaCell (DC) program are explained. In order to run all example steps, a Linux machine with Python and AmberTools is required. The DC executable is a 64bit ELF-binary; different executables can be provided upon request (m.zachmann@tum.de).

Preparation

DC requires coordinates (via .pdb-files) and a topology (via .top-files in DC-specific format). The topology-files can be converted from Amber-topologies (.prmtop-files) with the *Amber2Dynacell.py* script.

For this example, the protein was prepared in the following way: the dynamics of the protein-ligand complex (PDB-ID: 1BGQ) were simulated with Amber for 100ns at 300K in explicit solvent, and the trajectory was subsequently clustered. The representative structure of the highest populated cluster was extracted and defines the equilibrated holo complex. From this structure, all protein atoms were saved as *protein.pdb*. A corresponding Amber-topology was prepared in the file *protein.prmtop*.

Similarly, the ligand was separately simulated for 3x1000ns at 600K in explicit solvent to generate conformations of the macrocyclic ring. A representative conformation was saved without solvent as *ligand.pdb*, located in the *bs* folder. A corresponding Amber-topology was prepared in the file *ligand.prmtop*.

The Python-script for topology-file conversion requires access to AmberTools, and can be executed as follows:

For the protein:

```
./Amber2Dynacell.py -p protein.prmtop -o protein
```

and for the ligand:

```
./Amber2Dynacell.py -p ligand.prmtop -o ligand
```

where the Amber topology-file is specified via *-p*, and the output-file base-name via *-o*. Each command generates a DC-topology, named *protein.top* and *ligand.top*, respectively. Note, that if files with these names are already present in the output-folder, they can be overwritten with the *-O* option. The DC-topologies can now be used to run DC in the following broad sampling step.

Broad sampling

In this step, the desired number of random broad sampling ligand-poses is generated, by calling the DC executable with one argument:

```
./dynacell.64bit input.bs
```

where *input.bs* is a input text-file containing keyword-value pairs of input-options, and optional comments, introduced by a ";" character. The keyword dc_modus defines which module of DC is called, here it is set to "broadsampling". Protein and ligand .pdb-files are specified via pdbfile and ligandpdbfile, respectively. If a single .pdb-file is used for the complex, comprising both protein and ligand molecules, only the pdbfile keyword is used and ligandpdbfile should be erased. Similarly, DC-topologies are specified via topologyfile and ligandtopologyfile.

The generated poses are written as a .pdb-file to outputframefile. All poses will be written to a single file (separateOutFiles is “no”), separated by MODEL/ENDMDL entries. To keep the file size small, only the ligand atoms are written (writeWholePose is “no”), since the conformation of the protein doesn’t change during ligand-pose generation. Note that the input coordinates are printed as pose 0 for convenience. The log-output-file is specified via outputfile, which will list pose-energies and some structural information. It also contains a RMSD column of the ligand poses with respect to pose 0, where hydrogen atoms are not considered (RMSDwithHydrogens is “no”). Both output-files can be found in the *bs* folder.

The keyword ligandtype is set to “small_molecule”, indicating that the rotatable bonds must be manually defined in the bonds-section in the ligand topology-file *ligand.top*, via a “R” column next to the desired bonds. Note, that some bonds (e.g. bonds which are part of a ring, or “end”-bonds) are never rotated, and marking these with “R” is silently ignored. The total number of generated poses is given by numPoses. During broad sampling, atomic overlap is allowed, which will be removed in the following OPMD step, see below. The maximum allowed amount of overlap between protein and ligand atoms is specified via ProtLigOverlap (60% in this example). Similarly, the maximum allowed overlap within the ligand is given by LigLigOverlap (40% in this example). Internal ligand overlap can be especially created, when the ligand conformation is modified by randomly rotating around bonds. The sampled space is further defined by maxdistancefromprotein, giving the distance in nm that the ligand’s center of geometry (cog) is allowed to be away from at least one protein atom. Random rotations around bonds (marked with a “R”, see above) are enabled by setting rotationMode to “randomized”. Random rigid-body rotations of the ligand are enabled by setting rotate_ligand to “yes”. The cog of the ligand is randomly (translation_mode is “randomized”) placed within a sphere with radius translationDistance around the cog of the input coordinates, effectively defining the binding site. Finally, a maximum number of failed attempts per pose can be set (maxNrTries $\geq$ 0), to prevent DC from running forever, if no pose can be generated with the current settings.

A quick visual inspection of the generated broad sampling poses is recommended. A rough idea about the sampled space can be obtained by displaying all poses at once, together with the protein (e.g. with VMD or PyMOL). The cloud of ligand poses should fill the desired binding site sufficiently. Different settings for, e.g., ProtLigOverlap, maxdistancefromprotein and translationDistance can be tested to find appropriate values for the specific system.

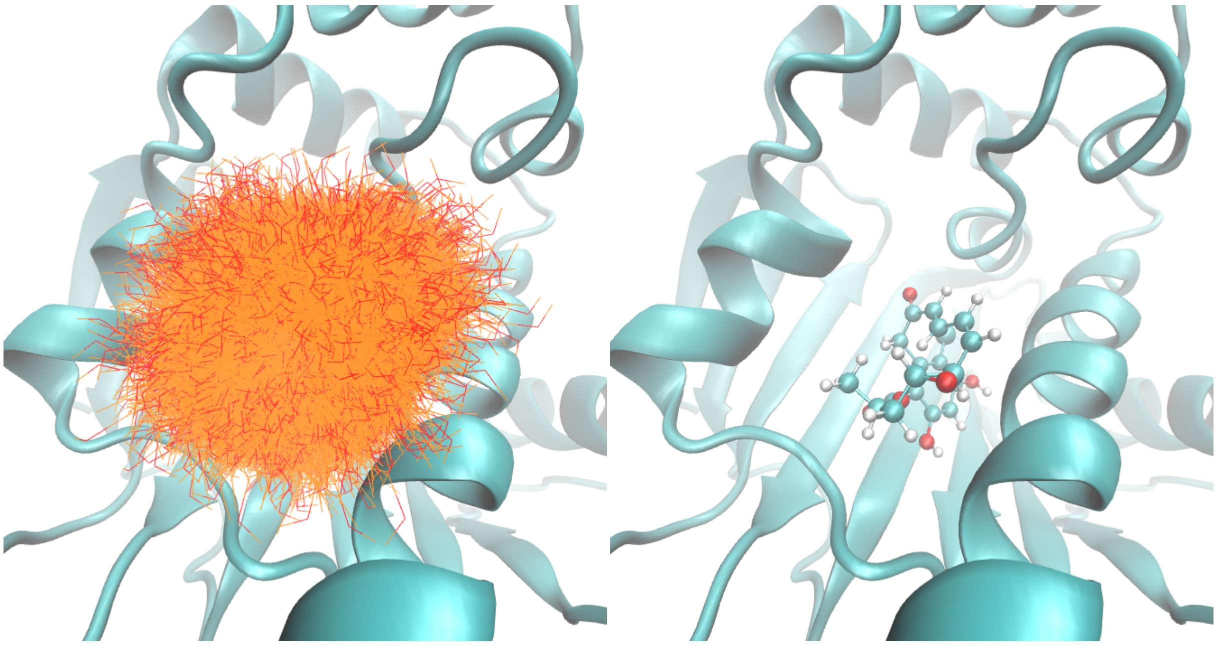
Figure 1. Left: 2000 broad sampling poses generated by DC displayed as orange lines. Right: The bound structure for reference.

As shown in **Figure 1**, the ligand poses fill up, and fuzzily resemble, the shape of the binding site. The allowed atomic overlap enabled sampling even in areas that are occupied by protein atoms.

The generated broad sampling poses were clustered to obtain a smaller set of diverse structures considered in the following refinement step. The further procedure is explained exemplary for one of these cluster representatives, named *lig.pdb* in the *md* folder. It corresponds to pose-number 1563 in the *bs.poses.pdb* file.

OPMD refinement

The molecular dynamics (MD) based refinement step applies non-bonded softcore (SC) potentials, described by a dimensionless parameter $\alpha \in [0,1]$, between ligand and protein atoms. This allows the simulation to run, even though there are overlapping atoms. These potentials are dynamically changed into normal ones ($\alpha=0.0$) during the simulation. It is again started by calling the DC executable with one argument:

./dynacell.64bit input.md

where *input.md* is a input text-file containing appropriate keywords. The dc_modus keyword must be set to “md”. The values for pdbfile, topologyfile and ligandtopologyfile are the same as for the broad sampling. The ligandpdbfile is one of the previously clustered broad sampling poses. Additionally, a text-file specifying the backbone .pdb-atom names within residues must be specified via backboneFile. This tells DC which atoms are considered backbone, side chain or not-peptide, should the need for such a distinction arise. The file *backboneNames.dat* located in the *md* folder explains the format.

The two main output-files are defined via outputfile and trajectoryfile, specifying the log- and the trajectory-file, respectively. The log-file contains information about the system at certain MD-steps, the frequency of which can be specified via outwrite. The trajectory-format can be either .pdb or the binary .dcd (determined by the file-name ending given as the trajectoryfile value), where the frequency of output-frames can be specified via trajwrite. To visualize the .dcd trajectory in a viewer, a topology must be loaded alongside the trajectory, e.g. in the form of the *complex.prmtop* file or as .pdb-file containing the entire system. Both output-files can be found in the *md* folder. The keyword nrsteps sets the maximum number of steps that the simulation will be running, if the SC-α parameter does not reach zero.

When applying restraints to the molecules, care must be taken so that the ligand and binding site are free to move, while keeping the rest of the protein stable. For that, the system can be divided into two regions (inner and outer), and different restraints can be applied in each. In this example, the inner region is defined as all atoms within a 1.2nm shell around the ligand-pose (restraintRegionLigandRadius), and will have no restraints (restraintAtomsInner is "none"). The outer region is defined as everything else, and harmonic restraints with a force constant of 1000kJ/(mol nm^2) are applied to backbone heavy-atoms of the protein, defined via restraintForceconstOuter and restraintAtomsOuter.

The AGBNP2 implicit solvent model is enabled with the keyword solvent. The necessary AGBNP2-parameters for the system are provided by the *AGBNP2.dat* file in the *md* folder, and recognized by the keyword AGBNPparaFile. The file was generated with the help of Schrödinger tools, mapping parameters to atoms of the system; see also the comments at the beginning of the file. In order to simulate significantly overlapping atoms within the framework of an implicit solvent, some modifications are necessary (solventScaling). This means coupling the set of atoms which are to be evaluated by the solvent model to the progression of the SC-potentials. This avoids unrealistic solvent-forces resulting from regions with closely interacting atoms. After the SC-α parameter reaches zero, the full system is normally solvated (solventScalingAlphaUp and solventScalingAlphaLow are "0.0").

The MD employs a Langevin thermostat (langevin is "yes" and frictionCoefficient is 0.5ps^{-1}) at 300K (temperature) and a time-step of 1fs (timestep). If the SC-α parameter reaches zero, the simulation runs for another 40000 steps (nrStepsAfterSCisDone) with normal potentials. Bonds involving hydrogen atoms are constrained with the SHAKE algorithm (constraints and constraintsType).

Finally, keywords detailing the exact nature of the SC-potentials have to be specified. During the simulation, the Lennard-Jones and Coulomb potentials (SCprogressionPots is "LJ_Q") should be dynamically changed via the SC-α parameter. These SC-potentials are applied between protein and ligand atoms (SCnbRegion is "ligandIntAct"). The SC-α itself is to be set via a SC-energy minimization (SCprogression is "optimize") at automatically determined intervals (SCautomaticFrequency is "yes"), until α=0.0. The starting value for the SC-α will be automatically chosen (SCinitialAlphaLJ and SCinitialAlphaQ are negative), according to the amount of initial overlap.

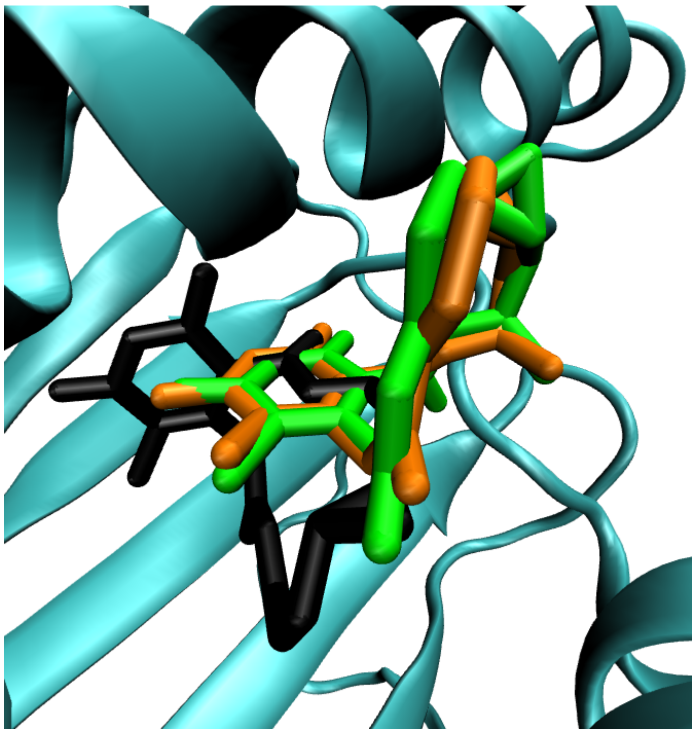

Figure 2. Movement of the ligand during the OPMD. Ligand in the first frame in black, and in the last frame in orange. The bound structure of the equilibrated holo complex is shown in green for reference.

In this example, the original broad sampling pose (black in **Figure 2**) exhibited a ligand heavy-atom RMSD of 5.24Å with respect to the bound reference. Throughout the OPMD simulation, the ligand moved into the correct position, as illustrated in **Figure 3**, and all atomic overlap was removed. At the end of the simulation (orange in **Figure 2**), the RMSD reduced to 0.69Å.

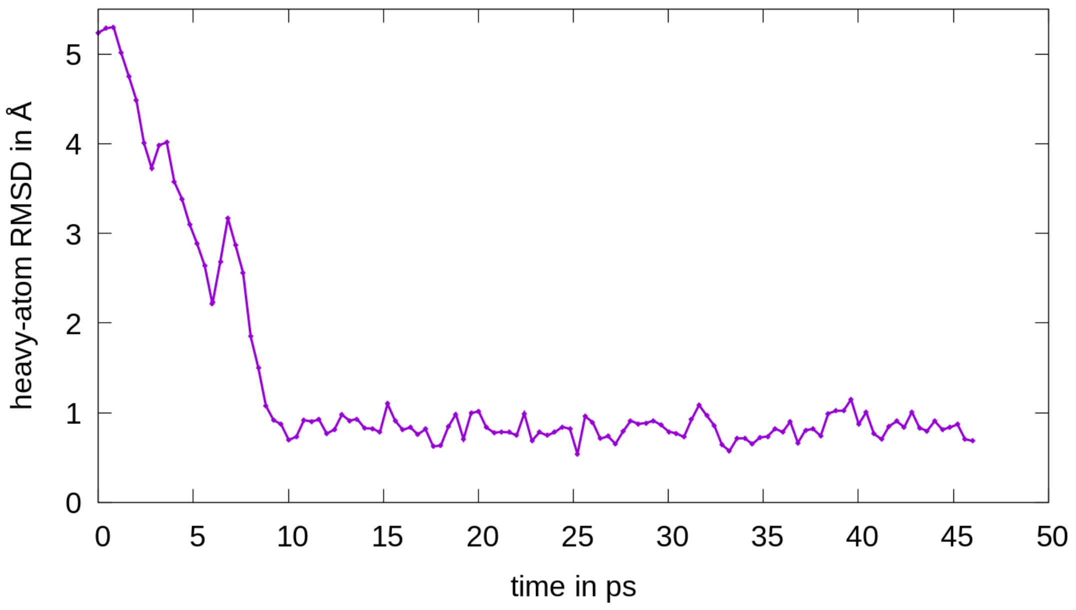

Figure 3. Ligand heavy-atom RMSD with respect to the bound reference throughout the OPMD refinement simulation.

Of course, not all broad sampling poses evolve like this, and mostly will not refine into the correct binding pose. Some poses do not refine to α=0.0 at all within the given simulation length. If the desired accuracy cannot be achieved, even with system-specific docking parameters, the number of generated poses should be increased. Longer simulation times after α reached 0.0, within DC or with an external MD-program like Amber, can also sometimes increase the pose quality.